Executive Summary

Assignment Overview:

In this project, we are to implement a fault-tolerant Key-Value store, ensuring continuous operation even when replicas fail, by incorporating the Paxos consensus algorithm. Key components of Paxos - Proposers, Acceptors, and Learners - must be effectively integrated to manage consensus among replicated servers, especially focusing on the event ordering. Also we have to configure the Acceptors to fail randomly, simulating real-world scenarios; this can be achieved through a timeout mechanism or random number generator forcing periodic failures. Additionally, you need to pre-populate the Key-Value with data, and then, using a client, demonstrate its working by performing a minimum of five operations each of PUTs, GETs, and DELETEs. This setup aims to test the system's resilience and the Paxos algorithm's effectiveness in maintaining a consistent state when there are multiple client interactions and internal failures.

Technical Impression:

My implementation works as follows:

The server side consists of a server(key-value store), Proposer, Acceptor, and Learner. The server envelops the given command into an operation object and then generates the proposal ID and calls the propose() method of the Proposer. The Proposer is responsible for coordinating the responses from the Acceptors. The Proposer calls the prepare() method of Acceptor with proposal ID and waits for the replies. The acceptor checks if the proposal ID that it has received is not less than the Promise ID (previously accepted proposal ID). If it is smaller then it sends the promise ID(which is higher than the proposal ID) in response, if not updates the promise ID to the given proposal ID and returns it. Proposer checks if more than the majority(floor(num_servers) / 2) of acceptors have replied or not. If not then the Consensus is not reached. Further from the returned replies checks if the majority of the acceptors have agreed to a single proposal ID or not. If not then consensus is not reached and it increments the proposal ID and goes through the prepare() method again. If the majority of acceptors have promised the same ID as the proposal ID then the accept() method of Acceptors is called, which replies with true if it can accept the proposal ID or false if it cannot accept it. Based on the given replies if a majority of Acceptors have returned true then Consensus is achieved and Proposer calls the learn() method of all Learners. The learners then apply the operation on the key-value store. Only the PUT and DELETE methods go through the PAXOS process. The GET method directly returns the value from the key-value store.

If “-f” parameter is set while running the server then it creates a 25% chance of the Acceptor failing. The Acceptor will still be running but it cannot be accessed for that run and acts like a simulated failure.

If 15000 is the given port parameter then the following ports will be used.

- Server0: 15000
- Server1: 15001
- Server2: 15002
- Server3: 15003
- Server4: 15004